|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 操作 | 操作人 | 备注 |
| 2021-6-2 | 创建文档 | 谷丛义 |  |
|  |  |  |  |
|  |  |  |  |

# 算法思路

## 概述算法步骤

先描述A\*算法的大致过程：

Openlist表示将要处理的邻居节点的集合

Closelist表示已经处理过的节点的集合，处理过就不在处理了

1.将初始节点放入到open列表中。

2.判断open列表。如果为空，则搜索失败。如果open列表中存在目标节点，则搜索成功。

3.从open列表中取出F值最小的节点作为当前节点，并将其加入到close列表中。

4.计算当前节点的相邻的所有可到达节点，生成一组子节点。对于每一个子节点：

1）如果该节点在close列表中，则跳过

2）如果该节点在open列表中，则检查其通过当前节点计算得到的F值是否更小，如果更小则更新其F值，并将其父节点设置为当前节点。

3）如果该节点不在open列表中，则将其加入到open列表，并计算F值，设置其父节点为当前节点。

5.转到2步骤

## 进一步解释

初始节点，目标节点，分别表示路径的起点和终点，相当于上图的绿色节点和红色节点  
F值，就是前面提到的启发式，每个节点都会被绑定一个F值  
F值是一个估计值，用F(n) = G(n) + H(n) 表示，其中G(n)表示由起点到节点n的预估消耗，H(n)表示节点n到终点的估计消耗。H(n)的计算方式有很多种，比如曼哈顿H(n) = x + y，或者欧几里得式H(n) = sqrt(x^2 + y^2)。本例中采用曼哈顿式。  
F(n)就表示由起点经过n节点到达终点的总消耗

# 具体如何计算请看下面的一个例子

用的消耗计算是x+y

G(n) ,H(n), F(n)

G(n)是当前节点到邻居节点距离

H(n)是相邻节点到目标节点的距离

F(n)= G(n)+H(n)

**初始：**

Openlist={A}

Closelist=null

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B}

Closelist={A}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | 1,8,9  A1 |  |  |  |  |  |  |  |
|  | 1,8,9  A2 | A | 1,6,7  B |  |  |  |  |  |  |
|  |  | 1,6,7  A3 |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C}

Closelist={A,B}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  | 1,7,8  B1 |  |  |  |  |  |  |
|  |  | A | B | 1,5,6  C |  |  |  |  |  |
|  |  |  | 1,5,6  B2 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D}

Closelist={A,B,C}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | 1,6,7  C1 |  |  |  |  |  |
|  |  | A | B | C | 1,4,5  D |  |  |  |  |
|  |  |  |  | 1,4,5  C2 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E}

Closelist={A,B,C,D}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | 1,5,6  D1 |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | 1,3,4  E |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F}

Closelist={A,B,C,D,E}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  | 1,4,5  E1 | E |  |  |  |  |
|  |  |  |  |  | 1,2,3  F |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G}

Closelist={A,B,C,D,E,F}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  | 1,3,4  F1 | F |  |  |  |  |
|  |  |  |  |  | 1,3,4  G |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H}

Closelist={A,B,C,D,E,F,G}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  |  |  |  |
|  |  |  |  | 1,4,5  G1 | G |  |  |  |  |
|  |  |  |  |  | 1,4,5  H |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H1,H2,I}

Closelist={A,B,C,D,E,F,G,H}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  |  |  |  |
|  |  |  |  |  | G |  |  |  |  |
|  |  |  |  | 1,5,6  H1 | H | 1,3,4  I |  |  |  |
|  |  |  |  |  | 1,5,6  H2 |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H1,H2,I1,J}

Closelist={A,B,C,D,E,F,G,H,I}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  |  |  |  |
|  |  |  |  |  | G |  |  |  |  |
|  |  |  |  |  | H | I | 1,2,3  J |  |  |
|  |  |  |  |  |  | 1,4,5  I1 |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H1,H2,I1,J1,J2,K}

Closelist={A,B,C,D,E,F,G,H,I,J}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  |  |  |  |
|  |  |  |  |  | G |  | 1,1,2  K |  |  |
|  |  |  |  |  | H | I | J | 1,3,4  J1 |  |
|  |  |  |  |  |  |  | 1,3,4  J2 |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H1,H2,I1,J1,J2,K1,L}

Closelist={A,B,C,D,E,F,G,H,I,J,K}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  | 1,0,1  L |  |  |
|  |  |  |  |  | G |  | K | 1,2,3  K1 |  |
|  |  |  |  |  | H | I | J |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Openlist={A1,A2,A3,B1,B2,C1,C2,D1,E1, F1,G1,H1,H2,I1,J1,J2,K1}

Closelist={A,B,C,D,E,F,G,H,I,J,K,L}

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  |  | F |  | L |  |  |
|  |  |  |  |  | G |  | K |  |  |
|  |  |  |  |  | H | I | J |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

找最小值的时候设置父节点

!openList.Contains(neighbourhoodNode)才设置父节点

即使找的图是

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
| L | M | N | O | P | E |  |  |  |  |
| K | J | I | H | G | F |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

这一轮没找到找openList最小的F进行进行，知道openList.Count==0 如果还没有有效路径 表示找不到，

否则反向追父节点

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | A | B | C | D |  |  |  |  |
|  |  |  |  |  | E |  |  |  |  |
|  |  |  |  | F1 | F |  | L |  |  |
|  |  |  |  | G1 | G |  | K |  |  |
|  |  |  |  | H1 | H | I | J |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

最后会追出一个基于当前成本的花费最小的路径

# 相关代码如下

public List<Vector2> GetRoad(InstanceMapNode start, InstanceMapNode end, bool needStop, ref bool bossRewardStop) {

if (start == null || start.IsSameNodePos(end)) {

return null;

}

List<InstanceMapNode> openList = new List<InstanceMapNode>();

List<InstanceMapNode> closeList = new List<InstanceMapNode>();

openList.Add(start);

while (openList.Count > 0) {

InstanceMapNode curNode = openList[0];

for (int i = 0; i < openList.Count; i++) {

if (openList[i].FCost <= curNode.FCost && openList[i].HCost < curNode.HCost) {

curNode = openList[i];

}

}

openList.Remove(curNode);

closeList.Add(curNode);

if (LTInstanceMapModel.Instance.BossRewardHashX > 0 || LTInstanceMapModel.Instance.BossRewardHashY > 0) {

//boss宝箱领取特殊处理方式

int temp = Mathf.Abs(curNode.X - LTInstanceMapModel.Instance.BossRewardHashX);

if (temp <= 1) {

temp = Mathf.Abs(curNode.Y - LTInstanceMapModel.Instance.BossRewardHashY);

if (temp <= 1) {

bossRewardStop = true;

return GeneratePath(start, curNode, true);

}

}

}

if (curNode.IsSameNodePos(end)) {

return GeneratePath(start, curNode);

}

var neighbourhoodNodeList = curNode.GetNeighbourhood();

for (var i = 0; i < neighbourhoodNodeList.Count; i++) {

var neighbourhoodNode = neighbourhoodNodeList[i];

if (neighbourhoodNode.Type == InstanceMapNode.NodeType.Wall || closeList.Contains(neighbourhoodNode))//避开墙和closelist中的节点

{

continue;

}

if (!neighbourhoodNode.IsSight)//被怪物控制的节点和未翻开的节点

{

continue;

}

if (!neighbourhoodNode.CanPass)//避开不能通行的格子

{

continue;

}

if (!needStop) {

if (!neighbourhoodNode.IsSameNodePos(end) && !IsCanStand(neighbourhoodNode)) {

continue;

}

}

if (!neighbourhoodNode.IsSameNodePos(end) && !IsRoleEmpty(neighbourhoodNode) && \_instanceType != InstanceType.Monopoly)//避开一切非空节点

{

continue;

}

int newCost = curNode.GCost + GetDistance(curNode, neighbourhoodNode);

if (newCost < neighbourhoodNode.GCost || !openList.Contains(neighbourhoodNode)) {

neighbourhoodNode.GCost = newCost;

neighbourhoodNode.HCost = GetDistance(neighbourhoodNode, end);

neighbourhoodNode.Parent = curNode;

if (!openList.Contains(neighbourhoodNode)) {

openList.Add(neighbourhoodNode);

}

}

}

}

return null;

}

# 路径生成

最后拿到curNode,也是end节点，找parent直到start形成一个列表，翻转一下就是路径